Session Code: ARC341

architecture & infrastructure

Next Generation Secure Computing Base - Overview and Drilldown

Keith Kaplan, Senior Security Developer
Ellen Cram, Lead Program Manager
Microsoft Corporation

PDC03

Make the connection

Microsoft

Agenda

- NGSCB overview
- Nexus fundamentals
- Writing an NGSCB agent
- Summary
- Q & A

Next Generation Secure Computing Base Defined

- Microsoft's Next-Generation Secure Computing Base (NGSCB) is a new security technology for the Microsoft Windows platform
  - Uses both hardware and software to protect data
  - Offers new kinds of security and privacy protections in an interconnected world

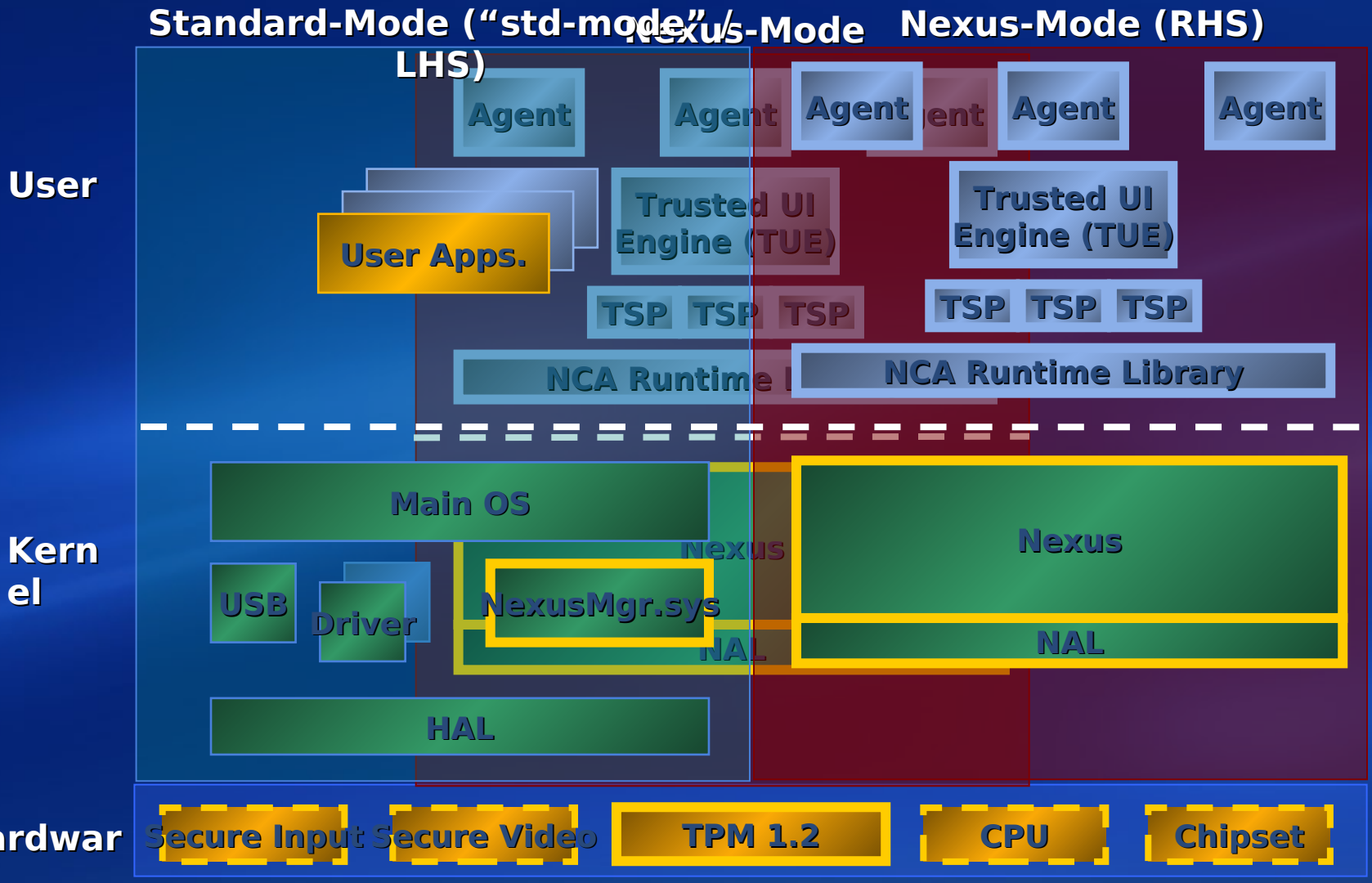
Threats Mitigated in V1

- Tampering with Data
  - Strong process isolation prevents rogue applications from changing our data or code while it is running
  - Sealed storage verifies the integrity of data when unsealing it
- Information Disclosure
  - Sealed storage prevents rogue applications from getting at your encrypted data
- Repudiation
  - Attestation enables you to verify that you are dealing with an application and machine configuration you trust
- Spoofing Identity
  - Secure path enables you to be sure that you're dealing with the real user, not an application spoofing the user

Version 1 Details

- Fully aligned with Longhorn
  - Ships as part of Longhorn
  - Betas and other releases in synch with and delivered with Longhorn's
- Focused on enterprise applications
- Example opportunities:
  - Document signing
  - Secure IM
  - Internal applications for viewing secure data
  - Secure email plug-in

NGSCB


Standard-Mode ("std-mode" / LHS)
Nexus-Mode
Nexus-Mode (RHS)
User
Kernel
Hardware
Agent
Agent
Agent
Agent
Agent
User Apps.
Trusted UI Engine (TUE)
Trusted UI Engine (TUE)
TSP TSP TSP
TSP TSP TSP
NCA Runtime Library
Main OS
Nexus
Nexus
USB
Driver
NexusMgr.sys
NAL
NAL
HAL
Secure Input
Secure Video
TPM 1.2
CPU
Chipset

Best of Both Worlds

- RHS = Security
  - In the presence of adversarial LHS code the system must not leak secrets
    - → The RHS must NOT rely on the LHS for security
- LHS = Richness and Compatibility
  - In the absence of LHS cooperation NGSCB does not run
    - → The RHS MUST rely on the LHS for stability and services

What Runs on the LHS

- Windows as you know it today
- Applications and drivers still run
- Viruses too
- Any software, with minor exceptions
  - The new hardware won't allow certain "bad" behaviors, e.g., code that:
    - Copies all of memory from one location to the next
    - Puts the CPU in real mode

Nexus Mode Environment

- Basic Operating System Functions
  - Process and Thread Loader/Manager
  - Memory Manager
  - I/O Manager
  - Security Reference Monitor
  - Interrupt handling/Hardware abstraction
- But not a complete Operating System
  - No File System
  - No Networking
  - No Kernel Mode/Privileged Device Drivers
  - No Direct X
  - No Scheduling
  - No…
- Kernel mode has no pluggables
  - All of the kernel loaded at boot and in the PCR

NGSCB Features

- All NGSCB-enabled application capabilities build off of four key features
  - Strong process isolation
  - Sealed storage
  - Secure path
  - Attestation
- The first three are needed to protect against malicious code
- Attestation breaks new ground in distributed computing
  - “Subjects” (software, machines, services) can be securely authenticated
  - This is separate from user authentication

Strong Process Isolatio

- Agents run in curtained memory
  - Not accessible by other agents
  - Not accessible by the standard Windows kernel
  - Not accessible by hardware DMA
- Enforced by NGSCB hardware and software
  - Hardware notifies Nexus of certain operations
  - Nexus arbitrates page tables, control registers, etc.

Secure Path

- Secure input
  - Secure session between device and Nexus
  - Protects both keyboard and mouse
  - USB for desktops, integrated input for laptops
- Secure output
  - Secure channel between graphics adaptor and Nexus

Sealed Storage

- Provides a method for encrypting data with a key rooted in the hardware
  - Sealed data can only be accessed by authenticated entities
  - Each Nexus generates a random keyset on first load
  - TPM chip on motherboard protects the Nexus keyset
  - Agents use Nexus facilities to seal (encrypt and sign) private data
  - The Nexus protects the key from any other agent/application, and the hardware prevents any other Nexus from gaining access to the key

Attestation

- When requested, the Nexus can prepare a chain that authenticates:
  - Agent by digest, signed by the Nexus
  - Nexus by digest, signed by the TPM
  - TPM by public key, signed by OEM or IT department
- The machine owner sets policy to control which forms of attestation each NCA or group of NCAs can use
- Secure communications agent provides higher-level services to agent developers
  - Open a secure channel to a service using a secure session key
  - Respond to an attestation challenge from the service based
    on user policy

“Booting” the Nexus

- The Nexus is like a kernel
- A kernel has to boot sometime
- The Nexus can boot any time
- It can shut down when it’s not needed (and restart later)

Shadow Process and Threads

- The Nexus has no scheduler
- LHS threads to call the right to load and run a RHS thread
- These LHS threads are part of the Agent's LHS shadow process
- Not getting scheduled again does not leak a secret
- Safe RHS synchronization primitives

Summary

- NGSCB ships as part of Longhorn
- NGSCB is a combination of
  - New hardware which creates a secure environment for…
  - …A new kernel, called the Nexus, which…
  - …Will run agents in a secure memory partition, and which…
  - …Will provide these agents with security services so that they can…
  - …Provide users with trustworthy computing
- Remember that:
  - When the Nexus is turned off, literally everything runs just like before
  - When the Nexus is on, the LHS runs very close to everything that ever ran
  - The Nexus makes no claims about what runs on the LHS
  - The hardware should run any Nexus, and give full function to any Nexus (with, at most, an admin step by the user)
  - The Nexus will run any software the user tells it to

Writing An Agent

Ellen Cram
Program Manager
Microsoft

PDC03
Make the connection

Writing NGSCB Agents

- Agents may be written in C or C++, using any compiler
  - Agents can be instantiated from managed or unmanaged code
- Once we have a RHS CLR, agents will be able to be written in any .Net language
  - The RHS CLR is planned to ship subsequently

Writing NGSCB Agents

- Two classes of RHS functions:
  - Functions enhanced by NGSCB
    - We use the hardware and protected environment to make these calls safer than they would be on standard Windows
  - Functions not protected by NGSCB
    - Indicated by a specific prefix
    - These functions are not any safer than an equivalent function in standard Windows
      - Developers must interact with these intelligently, such as encrypting data using sealed storage before writing it to disk
    - Our goal is to enhance these functions with NGSCB in future version

Types of Agents

- Application agents - stand-alone applications
  - The entire application runs on the RHS
  - Application agents are good for clients in multi-tier applications
  - Example: online banking client
- Component agents - components of a larger application
  - Most of the app runs on the LHS
    - Agents are used for specific trusted operations
  - A LHS proxy translates between COM or .Net and NGSCB IPC
  - Good for adding trusted features to existing Windows apps
  - Example: document signing component of a word processor

Agent Manifest

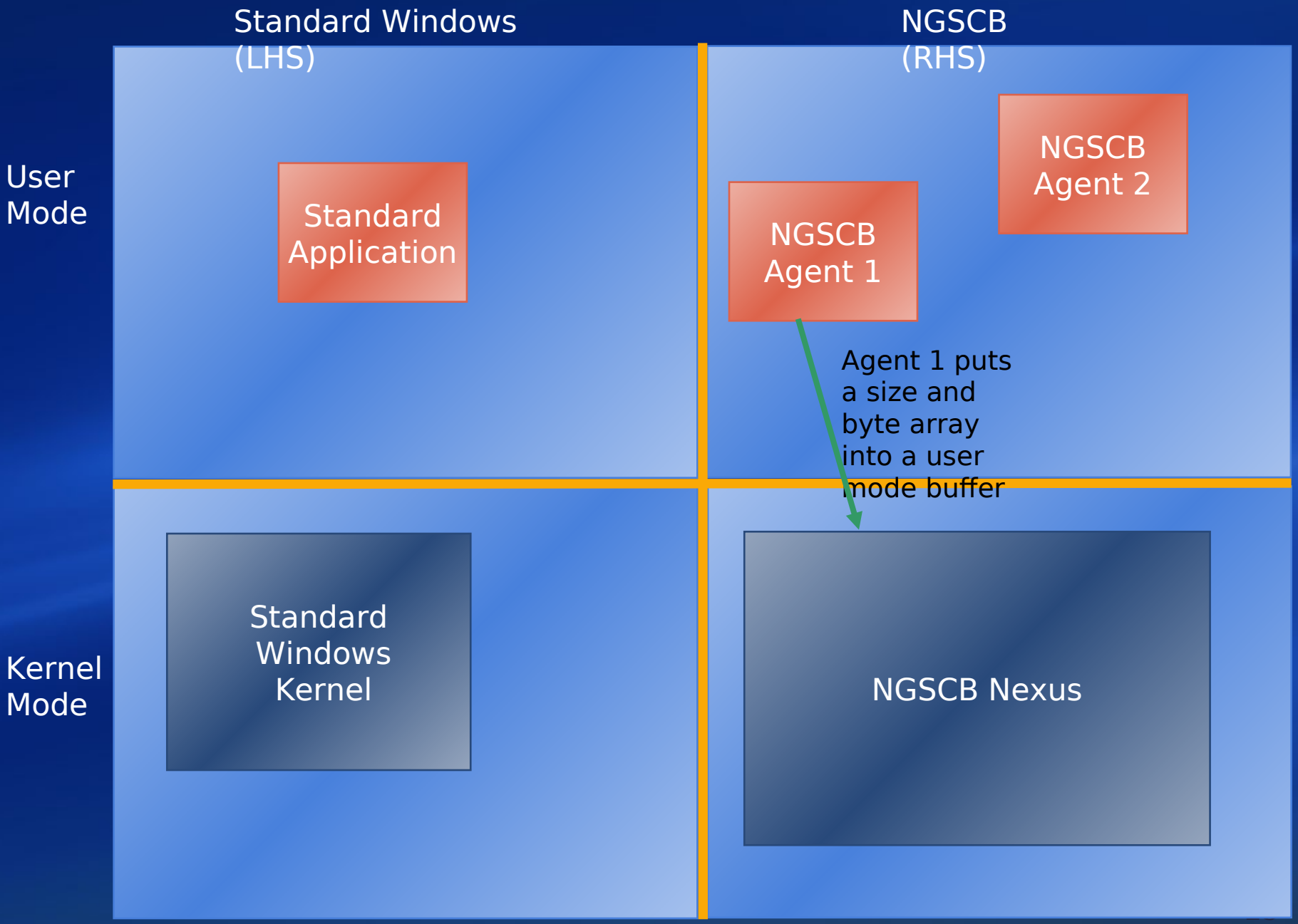
- Provides the information about an application that a machine user uses to determine if the app should run
- Signed XML document that defines:
  - Agent components
  - Agent properties
    - System requirements
      - Enforced by NGSCB
      - E.g. Debuggable = FALSE
    - Descriptive properties
      - Not interpreted nor enforced by the system
      - E.g. Version = 1.1.2.2
  - Agent policy requests
    - E.g. access to trusted output, write access to a counter, etc.
- XML schema is an NGSCB-specific extension to the standard Longhorn manifest
- Policy requests are not binding
  - Machine owner policy overrides manifest policy requests

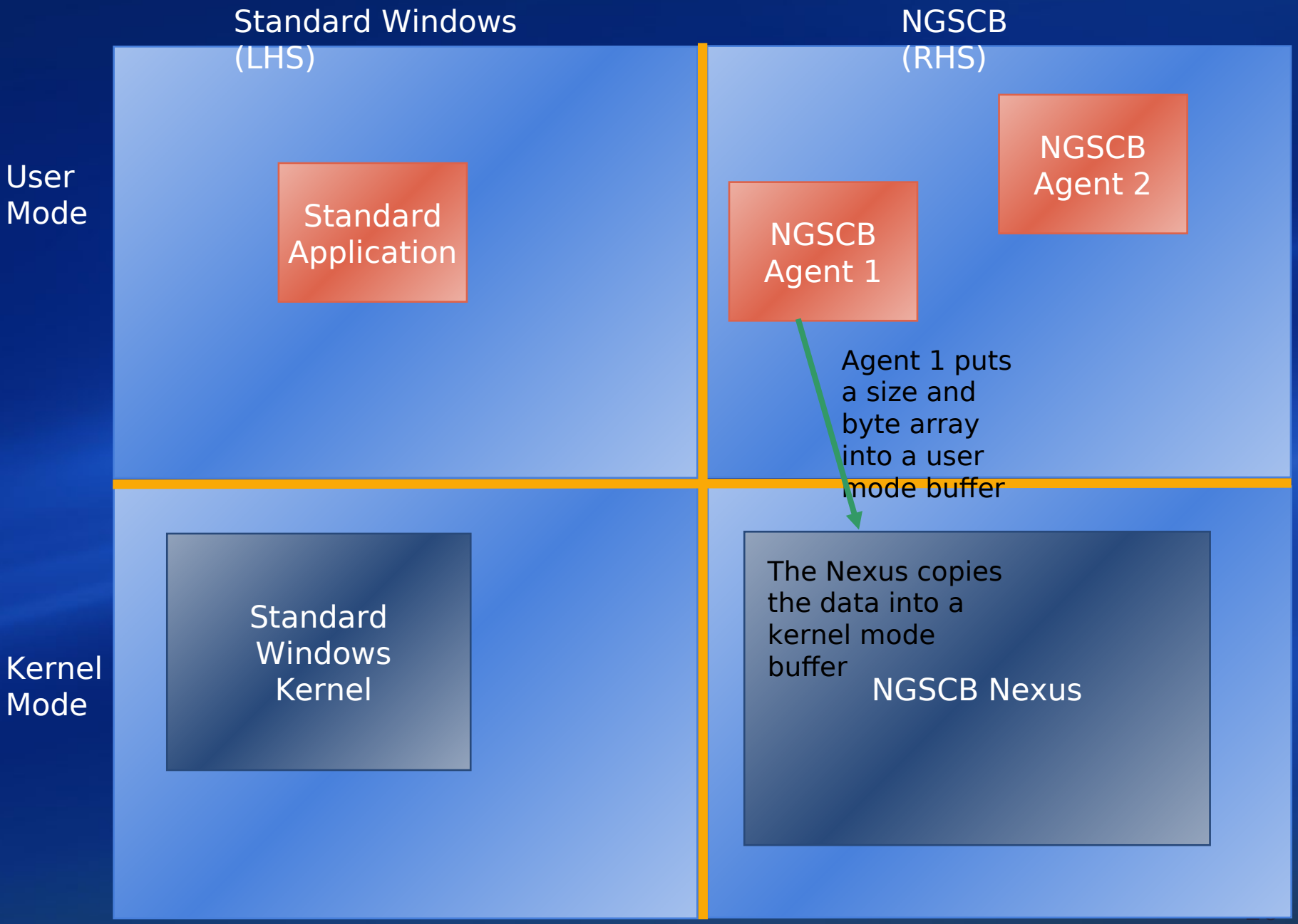
System Policy

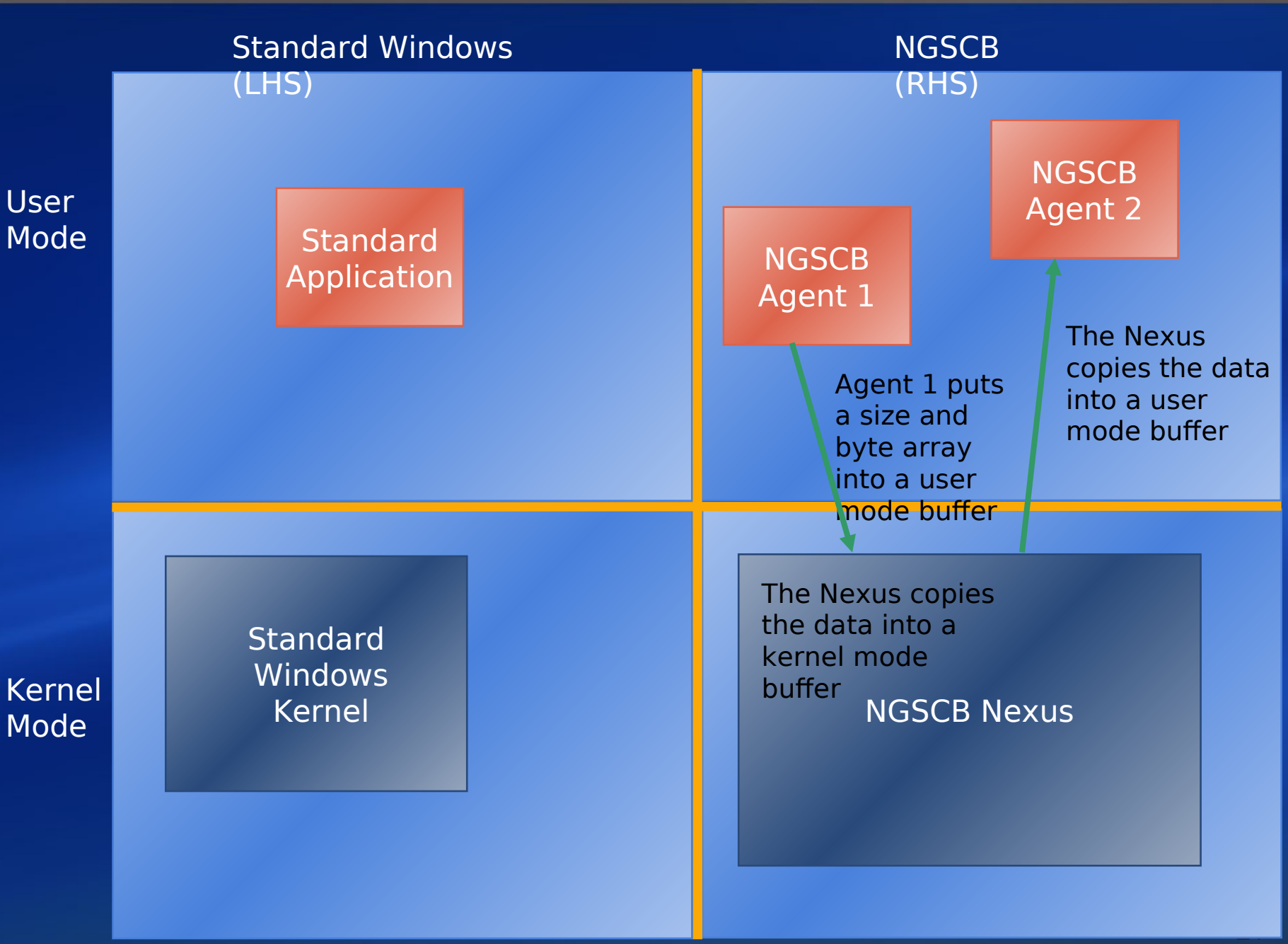
- Set by the machine owner
  - The owner may allow users the ability to override or extend
  - The owner may choose to delegate policy and trust decisions to a 3rd party
    - “Use Foo Org’s policies for any agent signed by Bar”
    - “Use my IT department’s policies for all agents”
- Expressed using signed XrML policy certificates
- Resources controlled by system policy include
  - Running an agent
  - Responding to an attestation challenge
  - Accessing a specific secret
  - Accessing NGSCB API sets (such as the network API)
  - Creating a child process
  - Accessing the TUE
- Policy is checked at run-time for every request
  - Some policy decisions are cached in the Nexus for performance reasons

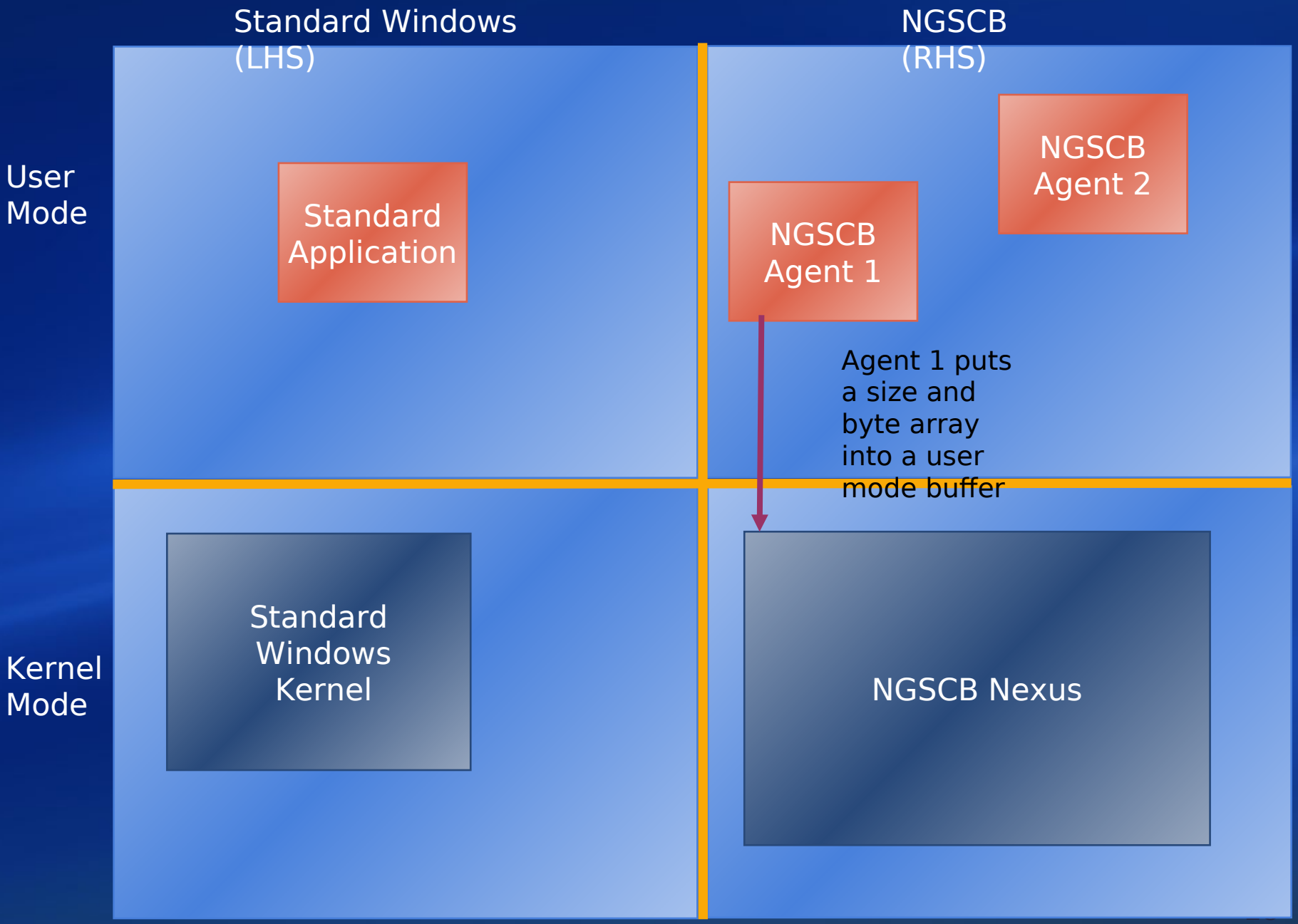
Factoring Agents

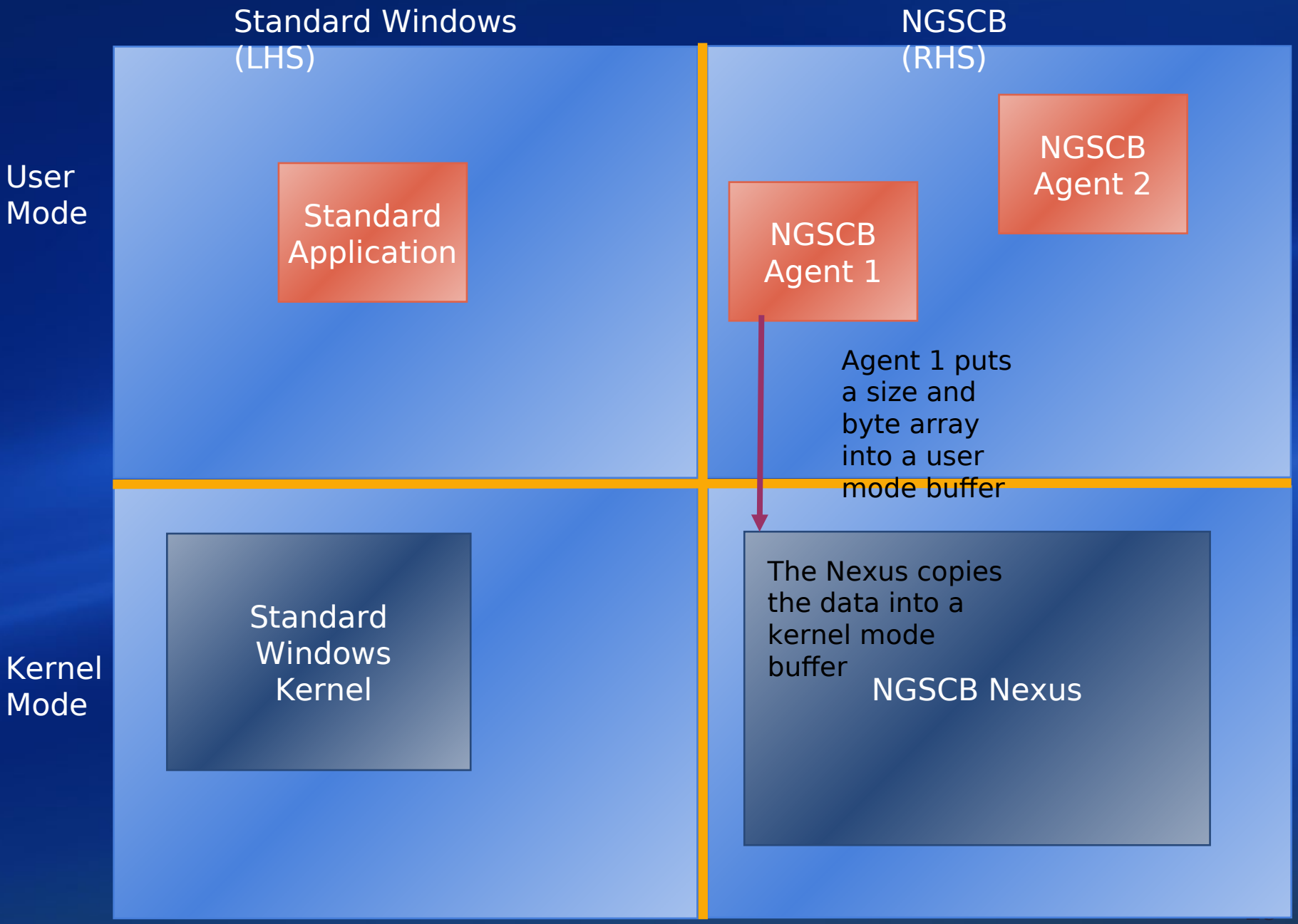
- Agents are monolithic - no DLLs
  - Code can be shared using statically-linked libraries
- Composition of agents is based on IPC
  - IPC is blocking and message-oriented
  - Agents and LHS processes can both use IPC
    - Agents can communicate with other agents
    - LHS applications can communicate with agents they start
  - Access to IPC is controlled by policy

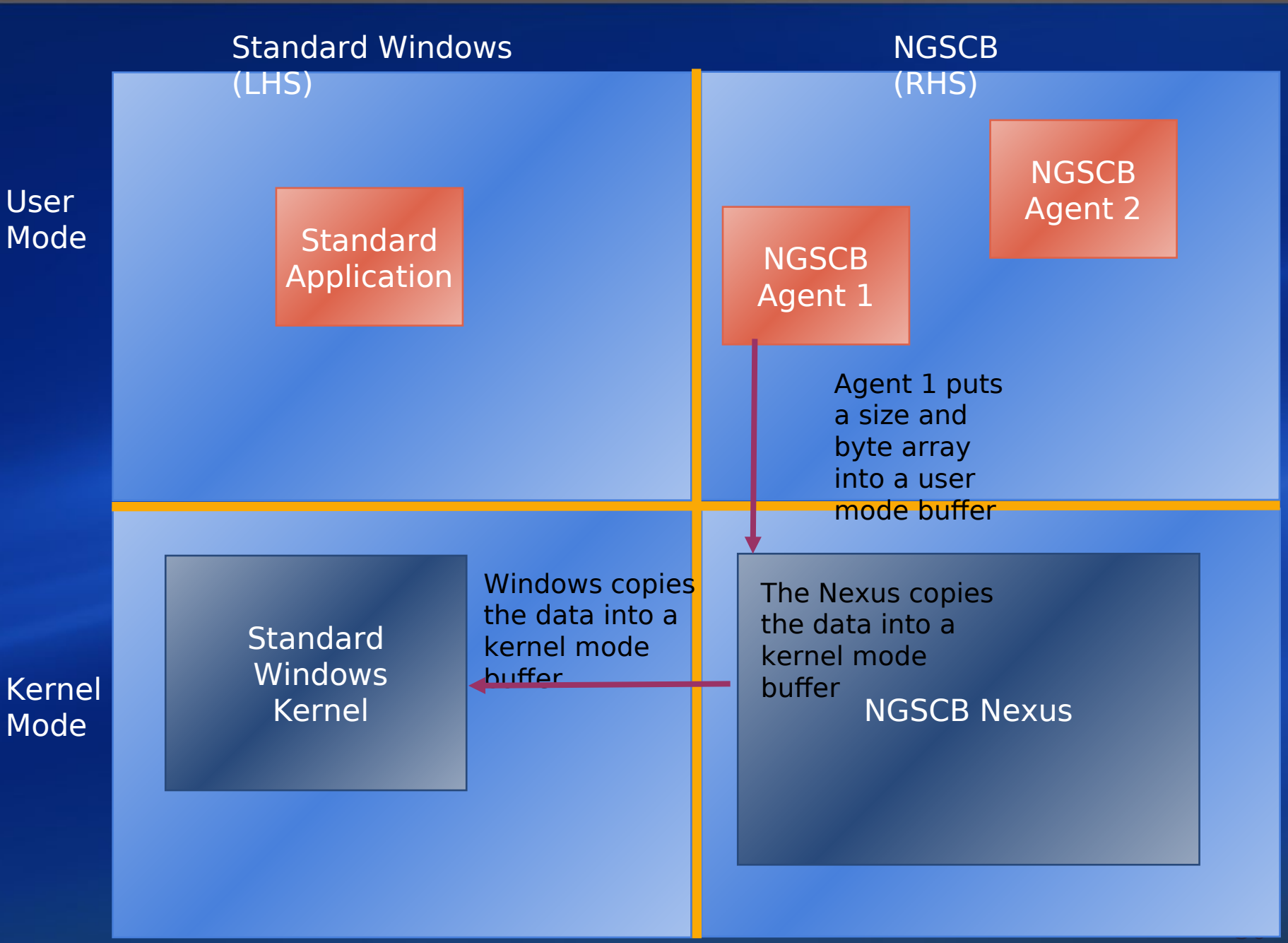

Standard Windows
(LHS)
NGSCB
(RHS)
User
Mode
Standard
Application
NGSCB
Agent 1
NGSCB
Agent 2
Agent 1 puts
a size and
byte array
into a user
mode buffer
Kernel
Mode
Standard
Windows
Kernel
NGSCB Nexus

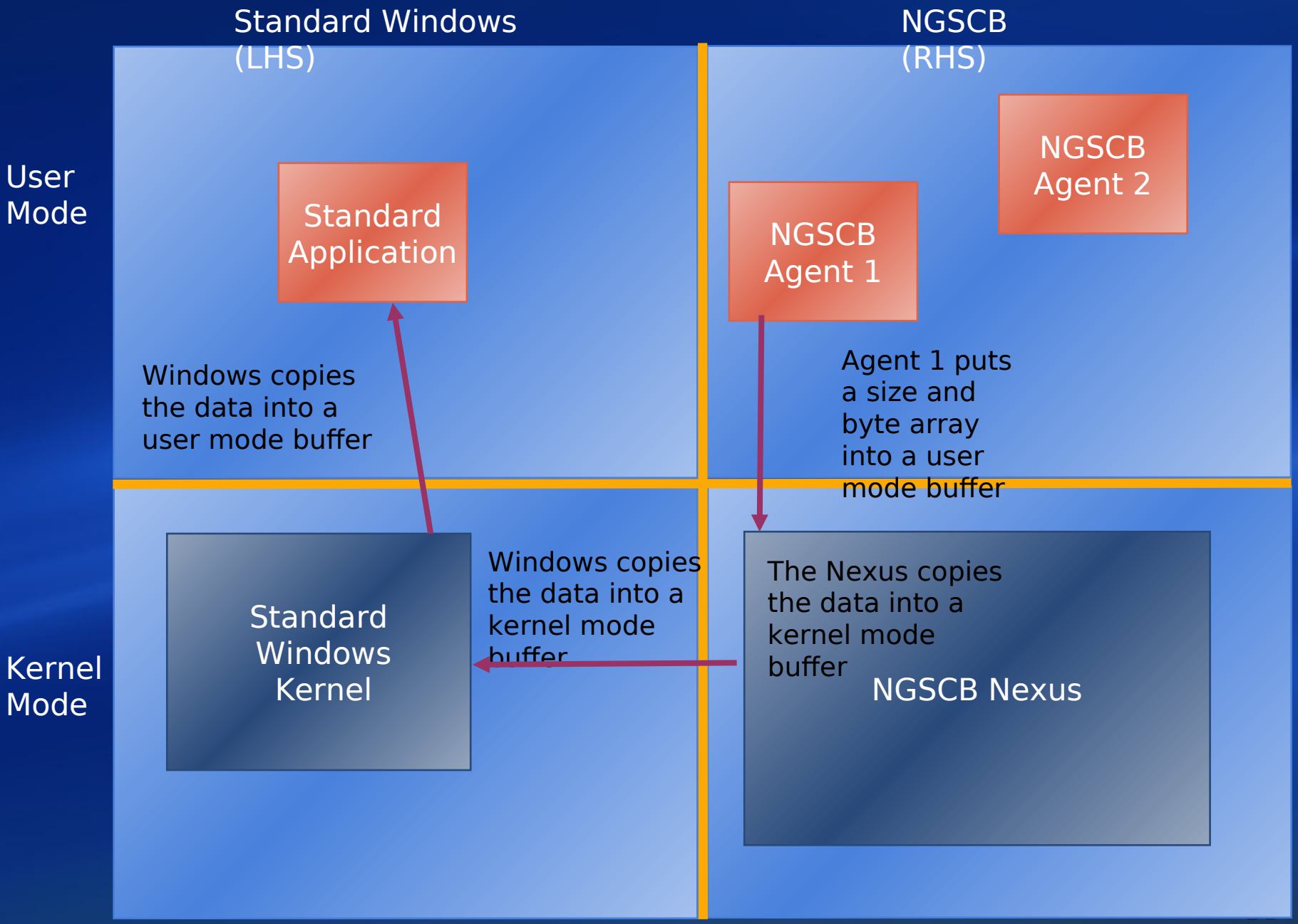

Standard Windows
(LHS)
NGSCB
(RHS)
User
Mode
Standard
Application
NGSCB
Agent 2
NGSCB
Agent 1
Agent 1 puts
a size and
byte array
into a user
mode buffer
Kernel
Mode
Standard
Windows
Kernel
The Nexus copies
the data into a
kernel mode
buffer
NGSCB Nexus


Standard Windows
(LHS)
NGSCB
(RHS)
User
Mode
Kernel
Mode
Standard
Application
NGSCB
Agent 1
NGSCB
Agent 2
Agent 1 puts
a size and
byte array
into a user
mode buffer
The Nexus
copies the data
into a user
mode buffer
Standard
Windows
Kernel
The Nexus copies
the data into a
kernel mode
buffer
NGSCB Nexus


Standard Windows
(LHS)
NGSCB
(RHS)
User
Mode
Standard
Application
NGSCB
Agent 1
NGSCB
Agent 2
Agent 1 puts
a size and
byte array
into a user
mode buffer
Kernel
Mode
Standard
Windows
Kernel
NGSCB Nexus


Standard Windows
(LHS)
NGSCB
(RHS)
User
Mode
Standard
Application
NGSCB
Agent 2
NGSCB
Agent 1
Agent 1 puts
a size and
byte array
into a user
mode buffer
Kernel
Mode
Standard
Windows
Kernel
The Nexus copies
the data into a
kernel mode
buffer
NGSCB Nexus


Standard Windows
(LHS)
NGSCB
(RHS)
User
Mode
Standard
Application
NGSCB
Agent 1
NGSCB
Agent 2
Agent 1 puts
a size and
byte array
into a user
mode buffer
Kernel
Mode
Standard
Windows
Kernel
Windows copies
the data into a
kernel mode
buffer
The Nexus copies
the data into a
kernel mode
buffer
NGSCB Nexus


Standard Windows
(LHS)
NGSCB
(RHS)
User
Mode
Standard
Application
NGSCB
Agent 1
NGSCB
Agent 2
Windows copies
the data into a
user mode buffer
Agent 1 puts
a size and
byte array
into a user
mode buffer
Kernel
Mode
Standard
Windows
Kernel
Windows copies
the data into a
kernel mode
buffer
The Nexus copies
the data into a
kernel mode
buffer
NGSCB Nexus

User Mode Debugging

- Agents are only debuggable if set in the manifest
  - Changing the manifest to enable debugging changes the agent's code identity
  - This change is reflected by attestation
- Debugging an agent really means debugging via LHS shadow process
  - We've redirected the functions to Get and Set Thread Context and Read and Write Process Memory
  - We've redirected RHS debug events to the LHS process
  - Thread control "just works"
- All well behaved debuggers that work with LHS processes will also with agents

What You Can Do Today

- The Longhorn release you receive here at PDC contains NGSCB developer preview
  - The Longhorn SDK also contains APIs for NGSCB
- The developer preview SDK is provided so that developers can understand the features and APIs we are providing
  - It does not demonstrate the security of NGSCB
- The NGSCB developer preview will enable you to prototype most applications you might write on NGSCB V1
  - The SDK may change before we RTM
- The developer preview includes a software emulator which simulates the NGSCB environment
  - You do not need new hardware to run it

NGSCB Developer Preview

- The developer preview supports
  - Creating an agent in Visual Studio
    - Debugging must be done on the command line at this point
  - Simulated Sealed Storage
  - Simulated Attestation
  - IPC
  - Standard Windows and CRT style APIs
- The developer preview does not provide
  - Secure Path
  - Strong Process Isolation

Summary

- NGSCB is made up of four key features
  - Strong process isolation
  - Sealed storage
  - Secure path
  - Attestation
- These features can be utilized either through stand-alone or componentized applications
- The NGSCB development environment is similar, although more constrained, than the standard Windows environment
  - Agents are monolithic - no DLLs
  - Composition of agents is based on IPC
- You can start experimenting with NGSCB APIs as part of the Longhorn SDK

Resources/Next Steps

- Go to our Hands On Lab here at PDC
  - NGSCB developers, testers and program managers are ready to answer your questions
- Study the SDK you received here at PDC
  - NGSCB is part of the Longhorn SDK
- Ask your hardware and software vendors what NGSCB-enabled components they will provide
- Visit our site and read the white papers and specs
  - http://www.microsoft.com/ngscb
- Send questions to our Q&A alias
  - ngscb_qa@microsoft.com
- Sign up for e-mail updates
  - Subscribe to the NGSCB information newsletter for ongoing updates. Send blank e-mail to:
    - wtpiinfo-subscribe@pens.tm500.com